階層的キーワードに基づくファイル管理のUNIX環境への適用

多田 知正 † 樋口 昌宏 ‡ 谷口 健一 †

† 大阪大学
大学院基礎工学研究科 情報数理系専攻

‡ 近畿大学
理工学部 電気工学科

我々は，ファイルシステムにおける階層的キーワードに基づくファイル管理手法を提案している．これは，ファイルに階層的に管理されたキーワードの集合を名前として付与することにより，柔軟なファイル操作を可能とするものである．本稿では，この手法のUNIX環境への適用を考える．既存のUNIXアプリケーションとの互換性を保つためにディレクトリとの併用を行うための機能について検討した．また，利便性を高めるために必要となる機能についても検討し，これらの機能を提供するプロトタイプシステムの実装を行った．

Hierarchical-Keyword-based File Naming Scheme in UNIX Environment

Harumasa Tada † Masahiro Higuchi ‡ Kenichi Taniguchi †

†Graduate School of Engineering Science
Osaka University

‡School of Science and Engineering
Kinki University

We have proposed a file naming scheme based on hierarchical keywords. In our naming scheme, the name of each file is a set of hierarchical keywords, which enables flexible file handling. In this paper, we adapted our naming scheme to the UNIX environment. For compatibility with existing UNIX applications, we provide a function to use directories and hierarchical keywords together. Moreover, we provide some functions which improve usability. We implemented a prototype system with these functions.

1. まえがき

従来のファイルシステムでは，ファイルは木構造のディレクトリによって管理されており，ユーザはパス名によってファイルを指定する．このようなファイルの管理手法を階層的名前付け (hierarchical naming) によるファイル管理という．現在，二次記憶装置の容量の増大に伴い，ファイルシステムの管理するファイルの数も膨大なものとなっている．ディレクトリ構造は深く複雑になり，目的のファイルを探し出すことが困難になりつつある．ファイルをより適切に分類し，より効率的な検索を行うために，様々なファイル管理手法が提案されている [1, 2, 3, 4]．我々は，このようなファイル管理手法の一つとして階層的キーワードに基づく名前付け (hierarchical-keyword-based naming) によるファイル管理 (以下，階層的キーワードファイル管理) を提案している [7]．階層的キーワードファイル管理は，ファイルの名前として階層的に管理された1つ以上のキーワードを付与するファイル管理手法である．[7] では，従来の階層的名前付けによるファイル管理の問題点と，階層的キーワードファイル管理のもたらす利点について議論している．また，UNIX上でファイル管理システムのプロトタイプを作成し，実験を行っている．

本稿では階層的キーワードファイル管理を既存のUNIX環境に導入する場合に必要となる機能について検討を行った．階層的キーワードファイル管理では，ディレクトリの代わりに階層的キーワー

ドを用いてファイルの分類を行う．しかし，UNIXにおける既存のアプリケーションは，ディレクトリの存在を前提として実装されている．例えばLaTeXのようなアプリケーションはカレントディレクトリに自動的にファイルを生成する．また描画ツールのようなGUIベースのアプリケーションでは，編集するファイルを選択するための機構を内部に持つことが一般的であり，これらはディレクトリ階層を辿る形で目的のファイルを選択するように実装されている，そこで，これらのアプリケーションを利用できるようにするため，階層的キーワードファイル管理と従来のディレクトリを併用することを考え，そのために必要な機能をプロトタイプ上に実装した．

また，以前に作成したプロトタイプを試験的に運用した結果に基づき，利便性を向上するため，階層的キーワード入力の支援機能を新たに導入し，複数ユーザによるファイル共有のための仕組みを変更した．

以降，2節では階層的キーワードファイル管理について説明する．3節では階層的キーワードファイル管理のUNIX環境への適用について述べ，4節ではシステムの利便性を向上するための機能について述べる．5節では作成したシステムの実装について述べる．最後に6節で結論を述べる．

2. 階層的キーワードファイル管理

ファイル管理とは，ファイルに対してユーザが行う管理作業全般をいう．すなわちファイルの生成，分類，検索，削除である．

ファイル管理手法はおもにファイルの分類と検索によって特徴づけられる．例えば，UNIXにおけるファイル管理では，階層型ディレクトリにファイルを格納することでファイルを分類し，ディレクトリを辿ることでファイルを検索する．ファイルの名前は，ディレクトリの階層構造に基づく階層的なパス名である．

以下では我々の提案している階層的キーワードファイル管理[7]について簡単に述べる．

2.1 階層的キーワード

階層的キーワードファイル管理では，ファイルをディレクトリに格納するかわりに，キーワードを付与することで分類を行う．各ファイルは1つのファイル名と1つ以上のキーワードを持つ．それぞれのキーワードは“//”で始まり，“/”によって区切られた任意の文字列である．キーワード間には親子関係が存在する．例えば，`//animal/dog`は`//animal`の子である．これによりキーワードの集合は階層構造を構成する．この階層構造をキーワード階層と呼び，それぞれのキーワードを階層的キーワード(Hierarchical Keyword: HK)と呼ぶ．

2.2 ファイルの指定

ファイルを指定するには，`//photo,//people,//animal/dog,fig1.jpg`のように，“,”で区切られたHKの非順序リストとファイル名を記述する．ファイル名はHKの非順序リストの最後に“,”で区切って書く*．

ファイルの指定は，その中のHKのリストにマッチする全てのファイルを指定する．HKのリストがファイルにマッチするとは，リスト中の各HKが，ファイルに付与されたHKの少なくとも一つにマッチすることをいう．HKは，それ自身およびその子孫にマッチする．例えば，`//sports`はそれ自身の他に，`//sports/soccer`や`//sports/skate/figure`等にもマッチする．

ファイルの指定にファイル名が含まれるとき，そのファイルの指定はHKのリストにマッチするファイルのなかでファイル名が一致したものを指定する．例えば，`//sports,//photo,fig1.jpg`は，`//sports,//photo`にマッチする`fig1.jpg`というファイル名をもつ全てのファイルを指定している．

2.3 ファイルの分類

階層的キーワードファイル管理においては，ファイルにHKを付与することで分類を行う．

例えば，「2000年の8月に撮った鶴の写真」を格納することを考える．階層的なファイル管理では，`/image/photo/2000/August/bird/crane`のような深いディレクトリを生成し，ファイルを格納することは利便性を損なうため通常行われない．これに対し，階層的キーワードファイル管理では，`//image/photo,//2000/August,//bird/crane`といったHKを付与することで詳細な分類が可能である．

階層的キーワードファイル管理において，ファイルに付与されたHKとファイル名の集合は，一つの名前空間を形成する．各ユーザはそれぞれ自分の名前空間とキーワード階層を持っており，自分の名前空間においては，自由にHKを生成して，ファイルに付与できる．あるユーザが自分の名前空間において生成，付与したHKは，他のユーザの名前空間には影響を与えない．

2.4 カレントキーワードリスト

従来のUNIXにおいてはカレントディレクトリの概念があり，ユーザは相対パスを用いてファイルを指定することができる．このような相対的なファイル指定を可能とするため，カレントディレクトリに相当するものとして，カレントキーワードリスト(Current Keyword List: CKL)を導入する．これは現在ユーザが注目しているファイル集合を指定するHKのリストである．

CKLはキーワードリストを“,”で始めることで参照される．CKLのみを指定するには“,”のみを書く．例えばCKLが“//a,//b”のとき“,//c,//d”は“//a,//b,//c,//d”を表し，“,”は“//a,//b”を表す．

*階層的キーワードとファイルの指定の書式は[7]から一部変更されている．

2.5 ファイルの検索

階層的キーワードファイル管理では，ユーザが目的のファイルに付与された HK を一部しか知らないとき，CKL を変更しながらファイルを検索する．最初に，既知の HK を CKL に加え，CKL にマッチするファイルのリストを閲覧する．ファイル数が多く，リストから選び出すことが難しければ，別の HK を CKL に加えることでマッチするファイルの数を減らす．リストに目的のファイルが含まれていなければ，CKL から HK を削除し，新たな別の HK を加える．このとき，ユーザが加えるべき HK を思い付かなければ，適切な HK が見つかるまでキーワード階層に沿って探索することになる．

3. UNIX 環境への適用

ここでは，階層的キーワードに基づくファイル管理を UNIX 環境に適用する際に必要となる機能について述べる．

3.1 ディレクトリとの併用

階層的キーワードファイル管理では，ファイルの分類の手段としてディレクトリを用いない．しかし UNIX における既存のアプリケーションは，ディレクトリの存在を前提としている．これらのアプリケーションを用いる際には，ディレクトリが必要となる．例えば LaTeX のようなアプリケーションはカレントディレクトリに自動的にファイルを生成する．また描画ツールのような GUI ベースのアプリケーションでは，編集するファイルを選択するための機構を内部に持つことが一般的である．このような機構はディレクトリ階層を辿る形で目的のファイルを選択するように実装されているため，階層的キーワードファイル管理では用いることはできない．これらのアプリケーションを階層的キーワードファイル管理に適応するように修正することは可能であるが，個々のアプリケーションに対応するには非常に手間がかかり，現実的ではない．

そこで本稿ではディレクトリと，階層的キーワードファイル管理を併用することを考える．ディレクトリの名前や階層構造は従来の UNIX と同様に構成できるとする．ファイルの指定は 2.2節で述べた HK のリストによる指定と，従来のディレクトリのパス名を用いた指定の両方が可能である．

3.2 ディレクトリの生成と削除

単純にディレクトリを導入した場合，ユーザは従来どおり階層構造のディレクトリにファイルを格納することでファイルの分類を行い，適切な HK を付与しない場合があり得る．そのようなファイルが増えると，ユーザは結局深く複雑なディレクトリ階層に依存することとなり，階層的キーワードファイル管理の利点は失われてしまう．これを防ぐため，ファイルは HK を用いて分類することとし，ディレクトリを必要に応じて自由に生成，削除できるようにするための仕組みを導入した．

ファイルの実体はディレクトリとは独立して管理されており，ディレクトリ中の全てのファイルは実体へのリンクである．ユーザはディレクトリが必要な場合，新しいディレクトリを生成し，その後 HK を用いて必要なファイル集合を検索し，得られたファイル集合の各ファイルへのリンクを生成したディレクトリに張るという作業を行う．以降では，ディレクトリ D に HK で検索したファイルのリンクを張ることを D にファイルをロードするという．

ファイルはディレクトリとは無関係に HK によって分類されるため，ディレクトリの名前や階層構造は単純なもので良い．頻繁に利用するファイルを簡単な名前のディレクトリにロードしておくことで素早くアクセスすることも可能である．また複数のディレクトリを生成することで，複数のファイル集合を扱うことができ，それらの間は通常のディレクトリ移動で行き来できる．また不要になったディレクトリを削除することにより，全体のファイル数が膨大になった場合でも，単純なディレクトリ木を維持できる．ディレクトリを削除してもファイルの実体は失われず，HK を用いて同じファイル集合を任意のディレクトリに再構成できる．

また，従来のディレクトリ階層に対して本手法を導入した場合，既存のディレクトリ中のファイルも，HK を付与することで分類でき，HK が付与されたファイルは，ディレクトリを削除しても失われない．

3.3 ディレクトリキーワードリスト

ディレクトリには通常ある作業に関連するファイルの集合が格納されている．CKL はユーザが注目している HK のリストであるから，ディレクトリを移動すると，それにともなって変わるのが望ましい．

そこで，各ディレクトリに HK のリストを関連づける．各ディレクトリに関連づけられた HK のリストをディレクトリキーワードリスト (Directory Keyword List: DKL) という．CKL は，カレントディレクトリの DKL とする．すなわち，カレントディレクトリを移動すると，CKL は，移動先の DKL に変更され，また CKL を変更すると，カレントディレクトリの DKL も同様に変更される．

[7] では，ファイルの指定にファイル名のみを用いた場合に，CKL が自動的に付与されるとしている．しかし，本手法では，ファイルの指定にファイル名のみを用いた場合には，カレントディレクトリのファイルが参照される．CKL を付与するには，`,main.tex` のようにファイル名の先頭に “,” を付ける必要がある．

3.4 ディレクトリとファイル集合

各ディレクトリには，そのディレクトリ中のファイルの集合とそのディレクトリの DKL にマッチするファイルの集合の 2 つのファイル集合が関連づけられている．DKL にマッチするファイルをディレクトリにロードした直後は，2 つの集合は一致しているが，その後 DKL が変更されると，これらは一致しなくなる．すなわち，ディレクトリの内容と DKL の間に矛盾が生じる場合がある．そこで，

- ディレクトリの内容を DKL にマッチするファイル集合に常に一致させる.
- ディレクトリの内容を変更せずそのままにしておく.

の 2 通りの方法が考えられる. 前者の方法をとる場合, DKL を変更すると, DKL にマッチするファイル集合が自動的にロードされることになるため, ユーザは明示的にロードを行う必要が無い. しかし, ディレクトリの内容と DKL が矛盾しないようにするため, mv コマンドを用いてディレクトリ間のファイルの移動を行うことはできない. 後者の方法では, ユーザが明示的にロードを行う必要があるが, ディレクトリ上でのすべてのファイル操作が可能である. また, ディレクトリ上のすべてのファイルを消去した後, 改めてロードすることにより, 前者の方法と同様に 2 つの集合を一致させることができる. さらに, ディレクトリ内の全てのファイルに DKL を付与するという形で 2 つの集合を一致させることも可能である. 現在の実装では, 後者の方法を採用している.

3.5 実行例

システムの提供する基本的なコマンドは表 1 の通りである. 従来の UNIX で提供されているファイル操作コマンドも含まれるが, 一部従来の UNIX と異なる動作をするものがある.

表 1 コマンド一覧

mkkw	HK の生成
rmkw	HK の消去
atkw	ファイルへの HK 付与
dtkw	ファイルからの HK 削除
akl	CKL への HK の追加
rkl	CKL からの HK の削除
ldf	ファイルのロード
chfn	ファイル名の変更
ls	ファイル名の一覧
mkdir	ディレクトリの生成
rmdir	ディレクトリの消去
cd	カレントディレクトリの変更
cp	ファイルの複製
mv	ファイルの移動
rm	ファイルの削除 (実体は残す)
rme	ファイルの実体の削除

実行例を図 1 と図 2に示す. 図 1は, 従来の UNIX のディレクトリにあるファイルに対して HK を付与する例である. プロンプトの {} 内は, カレントディレクトリを表す. 右側の () 内は, CKL を表しており, zsh の右プロンプトのような形式で表示されている. 4 行目と 5 行目の atkw でカレントディレクトリのすべてのファイルに対して, HK を付与している. その後, ディレクトリを消去している

図 1 実行例 (HK の付与)

```
tada{tada}>cd dps                                ()
tada{dps}>ls                                     ()
main.aux  main.dvi  main.log  main.tex
tada{dps}>atkw //society/IPSJ *                  ()
tada{dps}>atkw //SIG/DPS,//workshop *            ()
tada{dps}>rm *                                   ()
tada{dps}>cd ..                                  ()
tada{tada}>rmdir dps                             ()
```

図 2 実行例 (ファイルのロード)

```
tada{tada}>mkdir work
tada{tada}>cd work                               ()
tada{work}>akl //workshop                        ()
tada{work}>ls                              (workshop)
tada{work}>ls ,                            (workshop)
main.aux(3)  main.log(3)
main.dvi(3)  main.tex(3)
tada{work}>ldf                             (workshop)
Error: files with the same filename.
tada{work}>akl //SIG/DPS                         ()
tada{work}>ls ,                      (workshop,DPS)
main.aux  main.dvi  main.log  main.tex
tada{work}>ldf                       (workshop,DPS)
tada{work}>ls                        (workshop,DPS)
main.aux  main.dvi  main.log  main.tex
```

が, すでに HK が付与されているため, ファイルは失われない.

図 2は, 新しいディレクトリを作成し, そのディレクトリに, 検索したファイル集合をロードする例である. 3 行目の akl で CKL に HK を追加することで, ファイル集合の絞り込みを行っている. 4 行目の ls では, CKL に//workshop が入っているが, ファイルがロードされていないため, カレントディレクトリは空である. 5 行目の ls では, CKL の//workshop にマッチするファイルの一覧が表示される. ファイル名の後の "(3)" は, 同じファイル名のファイルが 3 つ存在することを表している. 8 行目の ldf によるファイルのロードはファイル名が重複するためにエラーとなっている. そこで, さらに HK を追加し, ファイル集合を絞り込んだ後に 13 行目で再びファイルをロードしている. ファイルをロードすることにより, 14 行目の ls ではカレントディレクトリのファイルの一覧が表示されている.

4. 利便性の向上

我々は, 前に作成した階層的キーワードファイル管理システムのプロトタイプ [7] を, 筆者の 1 人を含む 3 人のユーザで共有する 1000 の画像ファイルの集合に対して適用し, 画像ファイルの分類と検索を行う実験を行った. その結果, システムの利

図 3　キーワード入力支援 (1)

```
akl //animal/[Ctrl-U]
→ akl //animal/(cat)[Ctrl-U]
→ akl //animal/(dog)['/']
→ akl //animal/dog/[Ctrl-U]
→ akl //animal/dog/(bulldog)['p']
→ akl //animal/dog/p(oodle)[' ']
→ akl //animal/dog/poodle
```

図 4　キーワード入力支援 (2)

```
akl foo[Ctrl-U]
→ akl (//)foo(d)['t']
→ akl (//)foot[Ctrl-U]
→ akl (//movie/)foot(loose)[Ctrl-U]
→ akl (//sports/)foot(ball)[' ']
→ akl //sports/football
```

便性に関するいくつかの問題点が明らかになった．そこで，システムに以下のように改良を加えた．

4.1　キーワード入力支援

ファイルに HK を付与する場合やファイルを検索する場合には，ユーザは HK を入力する必要がある．HK は通常のキーワードと比べて，曖昧性を排除できるという利点がある半面，一般に通常のキーワードより長くなり，ユーザが正確に記憶しておくことは困難である．この場合ユーザはキーワード階層を辿ることで，目的の HK を見つけることができる．しかし HK のある一部分が分かっている場合でも，キーワード階層を根から順に探索することになり，大変手間がかかることになる．実際の運用を通じて，この HK の入力の手間が，利便性を大きく損なっていることがわかった．そこで，ユーザが階層的キーワードのある一部分を入力したときに，キーワード木を探索して，その文字列を含む HK の候補を表示する機能を追加した．

この機能は，HK を入力中に，特定のキー (現在の実装では "Ctrl-U") を入力することで起動する．キーを押すたびに，現在入力中の文字列を接頭語として含み，かつ "/" の数が等しい HK を候補として順に表示する．システムによって補完された部分は () で囲むことによって示される．図 3に例を示す．[ ] 内はユーザの入力したキーを表す．

入力中の文字列が "//" で始まっていない時は，キーを押すたびに，現在入力中の文字列が，その最後の要素の接頭語であるような HK を候補として表示する．図 4に例を示す．すなわち，一般的なシェルの補完機能のように階層構造を根から順に辿るだけではなく，入力中の文字列を含む HK をキーワード階層に因われずに探すことができる．

4.2　ファイル共有

階層的キーワードファイル管理においては，各ユーザは自分自身のファイルの名前空間とキーワード階層を所有する．[7] では，複数のユーザのファイルを互いに共有するため，他のユーザの名前空間を結合する方法を導入しており，これに基づくファイル共有の機能をプロトタイプ上に実装していた．しかし，実際に運用した結果，関連するファイルであっても，それぞれのユーザがファイルに付与するキーワードが異なるために，ファイルの検索がかえって困難になることがわかった．また，ファイル検索の結果，自分のファイルと他のユーザのファイルが一つの集合に含まれている場合，それぞれのファイルの所有者が分からないため，ファイル操作に支障をきたすことがわかった．

階層的キーワードファイル管理では，複数のユーザのファイルを一つの集合として扱うには，複数の名前空間を同時に参照する必要がある．このため，ファイルの共有を行うには，名前空間の結合が必要であった．しかし，ディレクトリとの併用により，各ユーザの名前空間を順に参照しながら，同じディレクトリにファイルをロードすることで，複数のユーザのファイルを一つのディレクトリ上で扱うことが可能となった．

そこで，複数の名前空間は同時に参照できないこととし，代わりに名前空間を切替える機構を導入した．現在参照している名前空間は DKL と同様にディレクトリごとに管理される．名前空間の切替えのために以下のコマンドを提供する．

- **chns** *user* (CHange Name Space)
  現在参照している名前空間をユーザ *user* のものに変更する．

5.　プロトタイプの実装

5.1　構成

本システムは，既存の UNIX オペレーティングシステム上に実装する形をとっており，通常のシェル上で実行される．ユーザはコマンドと，その引数としてのファイルを指定する．システムはファイルの指定を受け取り，それが HK を含む場合，指定されたファイルの絶対パスに変換し，コマンドと共にシェルに渡すという作業を行う．

5.2　キーワードの管理

本システムでは，キーワードの階層は，ディレクトリの階層として実装される．したがって，それぞれの階層的キーワードは実際のディレクトリに対応している．ただし，階層的キーワードに対応するディレクトリはシステムによって管理されており，ユーザは通常アクセスすることはない．階層的キーワードには，キーワード ID と呼ばれる一意な識別子が割り当てられている．

キーワードはテーブルによって管理される．これは，キーワード ID よりキーワードを得るためのものである．このテーブルをキーワードテーブルと呼ぶ．その例を表 2に示す．

キーワードがファイルに付与されると，そのファイルの実体へのシンボリックリンクが，キーワードに対応するディレクトリに配置される．このリンクの名前はファイル名とキーワード ID のリ

表 2　キーワードテーブル

ID	階層的キーワード
1	//image
2	//image/photo
3	//bird
4	//bird/crane
5	//machine
6	//machine/crane

ストより生成される．例えば，表 2 の状況において，//image/photo と//bird/crane という 2 つの HK を付与された fig.jpg というファイル名のファイルについて考える．この場合，リンクの名前は .#fig.jpg#2,4# となる．このリンクは，キーワード階層の根にあたるディレクトリを KWROOT とすると，KWROOT/image/photo と KWROOT/bird/crane の両方のディレクトリに配置される．

ファイル指定が与えられると，キーワードに対応するディレクトリがオープンされ，中のリンクが参照される．例えば，//image/photo,fig.jpg というファイル指定が与えられた場合，ディレクトリ KWROOT/image/photo がオープンされ，シンボリックリンク .#fig.jpg#2,4# が参照される．

5.3　ディレクトリの管理

本システムでは，不要になったディレクトリをいつでも削除できるようにするため，ファイルの実体はシステムによって管理される別のディレクトリに格納される．このディレクトリを実体ディレクトリと呼ぶ．ユーザは通常実体ディレクトリに直接アクセスすることはない．ユーザの扱うディレクトリの中のファイルは，原則としてすべてシンボリックリンクである．ただし新しく生成されたファイルで，キーワードを付与されていない状態では，ファイルの実体はそれが生成されたディレクトリにそのまま置かれている．ファイルにキーワードが付与された時点で，ファイルの実体は実体ディレクトリに移動され，ファイルは実体へのシンボリックリンクに置き換えられる．

ディレクトリは，DKL を保持するためのファイルを持つ．システムは，ユーザがディレクトリを移動すると，このファイルを参照して CKL を設定する．

6.　あとがき

本稿では，我々の提案している階層的キーワードに基づくファイル管理を UNIX 環境で適用する際に必要な機能について検討し，通常のディレクトリとの併用のための仕組みを導入した．また，利便性の向上のために，キーワードの付与支援機構を新たに導入し，ファイル共有の際の名前空間の扱い方を変更した．

このようなシステムの機能については，実際に利用していく中で発生するユーザの要求や不満をフィードバックしていくことが重要である．今後，作成したシステムの運用を通じて現在の仕様の問題点を抽出し，システムの再設計を行っていきたいと考えている．

References

[1] B. C. Neumann, "The Prospero filesystem: A Global File System Based on the Virtual System Model", Proc. USENIX Workshop on File Systems. May 1992.

[2] S. Sechrest and M. McClennen, "Blending hierarchical and attribute-based file naming", IEEE Proc. of the 12th IEEE Intl. Conf. on Distributed Computing Systems, pp.572-580, Yokohama, Japan, Jun 1992.

[3] D.K. Gifford, P. Jouvelot, M. A. Shedon and J. W. O'Toole, "Semantic File Systems", ACM Proc. of the 13th ACM Symp. on Operating Systems Principles, pp.16-25, Pacific Grove, CA, Oct 1991.

[4] E. Freeman and D. Gelernter, "Lifestreams: A Strage Model for Personal Data", ACM SIGMOD Record, vol.25, No.1, pp.80-86, Mar 1996.

[5] A. S. Tanenbaum, "Distributed Operating System", Prentice Hall, ISBN 0-13-588187-0, 1992.

[6] D. B. Terry, "Distributed Name Servers: Naming and Caching in Large Distributed Computing Environments", Technical Report CSL-85-1, Xerox Palo Alto Research Center, Feb 1984.

[7] H. Tada, N. Todoroki, K. Fukui, and M. Higuchi. "A Hierarchical-Keyword-based Naming Scheme in File Systems", 情報処理学会論文誌 Vol.42, No.9 (採録予定).